PERF - UM AMBIENTE DE DESENVOLVIMENTO VOLTADO PARA TESTES TEMPORAIS DE SOFTWARE

João Alexandre Góes e Douglas P. B. Renaux
Laboratório de Informática em Telecomunicações - LIT
Centro Federal de Educação Tecnológica do Paraná
Av. Sete de Setembro, 3165 - 80230-901 - Curitiba-PR
goes@cefetpr.br, douglas@cpgei.cefetpr.br
http://www.lit.cpdtt.cefetpr.br

RESUMO

Ferramentas de apoio os teste de restrições temporais são muitos úteis no desenvolvimento de Sistemas em Tempo Real. Apesar de sua importância, são raras as ferramentas existentes voltadas para este segmento. Assim, foi desenvolvido o ambiente PERF, composto de várias ferramentas, para desenvolvimento e avaliação de Sistemas em Tempo Real. Este artigo apresenta o ambiente PERF, seu atual estágio de desenvolvimento e a ferramenta de estimação de tempo de execução integrada ao ambiente, que tem o objetivo de, através de análise estática, estimar os tempos de execução de um sistema computacional.

ABSTRACT

Tools providing support for timing requirements tests are usefult in the development cycle of Real-Time Systems. Despite its importance, tools for this segment are rare. Hence, PERF was developed as an environment, compounded of a set of tools, that supports the development and analysis of Real-Time Systems. This paper describes PERF, its current development stage and the execution time analysis tool, whose aim is, through static analysis, estimate the execution time of a computational system.

1 INTRODUÇÃO

PERF é um projeto de longa duração visando o desenvolvimento de um ambiente que integre um conjunto de ferramentas que apóiem o desenvolvedor de sistemas em tempo real a estimar, medir e caracterizar o comportamento temporal de um sistema em tempo real [Renaux 99, Góes 01, Góes 01a]. Este projeto está sendo desenvolvido no LIT (Laboratório de Informática em Telecomunicações) do CEFET-PR. Este artigo apresenta o estágio atual de desenvolvimento do ambiente PERF: a ferramenta de análise estática de código-objeto e estimação temporal.

2 AMBIENTE PERF

O ambiente é fundamentado sobre um Núcleo Operacional (Figura 1), que é responsável pela gerência de todo o programa e não se altera devido aos diferentes cenários de desenvolvimento que no PERF podem ser desenvolvidos. Um cenário de desenvolvimento é um conjunto de ferramentas necessárias para se desenvolver um sistema computacional para uma plataforma de *hardware*. O Projeto Ativo é o elemento que mantém o conjunto de arquivos-fonte e toda a informação referente ao projeto em uso. O projeto ativo controla um conjunto de ferramentas, já implementadas, cada uma com uma função específica:

- *Editor, compilador e ligador*: provê no próprio ambiente PERF o controle de todo o processo de criação de programas.
- *Engenharia reversa*: extrai informações contidas nos arquivos-objeto (seções, funções, referências, relacionamento linha de código-fonte e endereço de seção, etc) e armazena-as, em um formato padronizado, em uma base de dados comum às ferramentas.
- *Base de dados*: mantém as informações sobre o projeto, como por exemplo, as seções dos arquivos-objeto, funções, registros de relocação, entre outras. Esta base de dados segue um padrão conhecido e pode ser acessada por todas as ferramentas do PERF.
- *Ferramenta de estimação de tempo*: determina os limites de tempo de execução de funções através de técnicas de estimação.
- *Sistema Operacional*: determina quais são os processos existentes no programa.
- *Processador*: modela as instruções e o comportamento temporal de processadores.

A funcionalidade de cada ferramenta é complementada com módulos externos (*plugins*) que contêm funcionalidades específicas para um cenário de desenvolvimento. Um *plugin* é uma biblioteca de vínculo dinâmico (dll - *dynamic load library*) resolvida em tempo de execução pelo PERF, e cada ferramenta pode carregar um *plugin*, com exceção da ferramenta de compilação, que pode car-

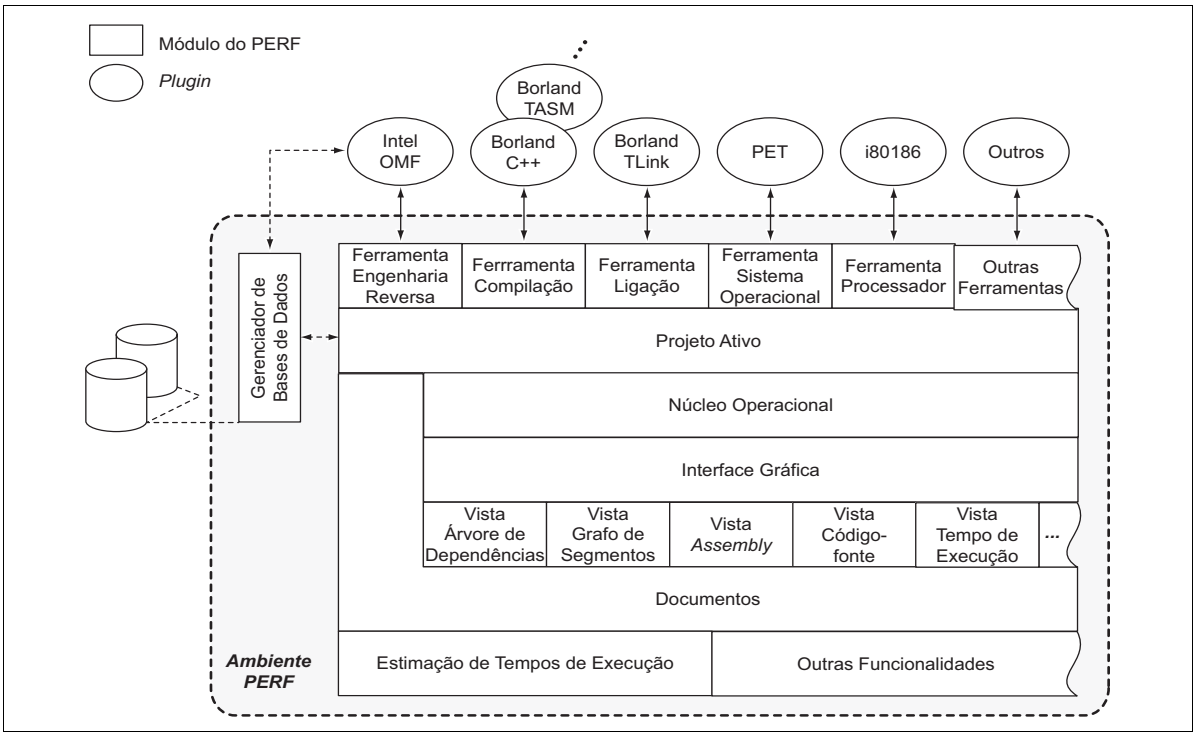

Figura 1 - Arquitetura modular do Ambiente PERF

regar um conjunto de *plugins*. Um *plugin* pode ser criado por qualquer compilador, desde que mantenha rigorosamente a interface definida pelo PERF.

A independência existente entre o conjunto de ferramentas e o conjunto de *plugins* permite que o ambiente acomode vários cenários de desenvolvimento diferentes. Por exemplo, o PERF permite o desenvolvimento de sistemas utilizando a plataforma 80186, usando os *plugins* Intel OMF, Borland C++, Borland Linker, PET e i80186; ou então permitir o desenvolvimento de sistemas utilizando a plataforma PowerPC 860, usando, por exemplo, os *plugins* ELF, Gnu Gcc, Gnu Linker, MPC860, entre outros. Existe neste trabalho a preocupação de que tanto o ambiente quanto as ferramentas sejam modulares e intercambiáveis, de forma que uma futura troca de arquitetura de microprocessadores implique apenas o modelamento de novos módulos do PERF.

A interface gráfica é responsável por gerenciar toda a interação de janelas, diálogos, menus e barras de ferramentas existentes no ambiente. Os resultados são apresentados em vistas (*views*), responsáveis por apresentar um tipo diferente de informação. Por exemplo, uma vista apresenta o código-fonte de uma função, outra vista apresenta o código em *Assembly*, enquanto outra apresenta o grafo de fluxo de controle.

As informações geradas pelo ambiente e pelos *plugins* são gravadas em uma base de dados no formato Microsoft Access, controlada por um gerenciador, de forma que todos os componentes do PERF, e até mesmo outros programas, possam acessá-las.

3 FUNCIONALIDADE

Todas as operações no PERF são controladas através de uma interface gráfica única (Figura 2), que é semelhante à utilizada na maioria dos ambientes de desenvolvimento atualmente em uso, e composta de quatro regiões: A região superior comporta os menus e a barra de ferramentas; a região à esquerda apresenta a área de trabalho, que contém a listagem de todos os arquivos-fonte do projeto e também todas as funções encontradas nos arquivos-objeto; a região inferior destina-se a apresentação de mensagens de erro, orientação ao usuário e principalmente a apresentação das informações provenientes dos compiladores e ligadores; e o restante da interface é destinada a mostrar janelas, contendo informações sobre o projeto, solicitadas pelo usuário.

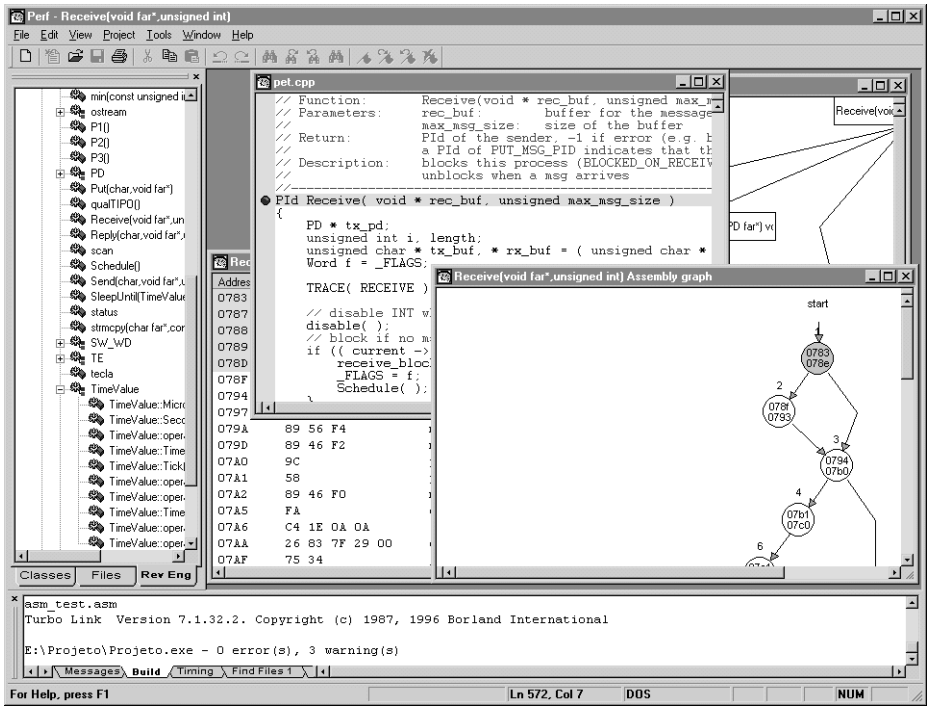

Figura 2 - Interface gráfica do PERF

3.1 Edição, Compilação e Ligação

O processo edição-compilação-ligação adotado pelo ambiente PERF é semelhante ao utilizado pela maioria dos compiladores comerciais. O projeto mantém uma lista de todos os arquivos-fonte necessários para gerar o programa executável, e os arquivos podem ser editados no ambiente através de um editor de texto, com suporte à sintaxe da linguagem de programação, próprio.

Após a edição dos arquivos-fonte, o ambiente PERF pode iniciar o processo de compilação dos arquivos a fim de gerar os respectivos arquivos-objeto. Todas as dependências existentes nos arquivos-fonte são verificadas com o objetivo de inserir automaticamente arquivos apontados por diretivas de inclusão (*includes*). Após a compilação, os arquivos-objeto gerados são utilizados pelo ligador para gerar o programa executável final, e também pelo processo de engenharia reversa, cujo objetivo é extrair as informações contidas e armazená-las na base de dados.

Os compiladores e ligadores são, essencialmente, programas de console, ou seja, não precisam de interface gráfica, os parâmetros de entrada são dados em linha de comando e os resultados apresentam-se em arquivos e em linhas de mensagens. Isto permite que os resultados apresentados pelos compiladores possam ser apresentados na interface gráfica do ambiente PERF.

3.2 Engenharia Reversa dos Arquivos-objeto

Uma das características do ambiente PERF é realizar o processo de estimação de tempo de execução de forma a ser independente de compilador, e isto implica utilização dos arquivos-objeto para a retirada das informações referentes às funções contidas no projeto. Apesar dos arquivos-objeto conterem informações semelhantes, estas informações são dependentes do formato do arquivo-objeto e, para que o requisito da modularidade seja atendido e para que todos os módulos possam acessar estas informações, elas são convertidas e inseridas em uma base de dados de formato conhecido. Bibliotecas de funções também são arquivos-objeto e, portanto, seu conteúdo também é convertido para a base de dados.

3.3 Estimação dos Limites de Tempo de Execução

A ferramenta de estimação de tempo de execução determina os limites de tempo de execução de programas: WCET (*Worst-Case Execution Time*), BCET (*Best-Case Execution Time*) e TCET (*Typical-Case Execution Time*).

Existem diversas abordagens para se determinar os limites de tempo de execução de programas [Puschner 00]. Algumas fazem a análise baseada em código-fonte: procurar caminhos existentes no código-fonte; para cada caminho do código-fonte determinar o caminho do código de máquina equivalente; e determinar o tempo de execução destes caminhos [Park 91, Stoyenko 91, Lim 94, Kawamura 00]. Outras abordagens analisam diretamente o código de máquina existente no programa executável [Li 95, Cinderella 01, Engblom 99]. O PERF adota a abordagem de determinar os caminhos de execução existentes no código de máquina, que é aquele que realmente estará sendo executado na plataforma alvo, utilizando informações existentes nos arquivos-objeto.

A determinação do tempo de execução de um programa é feito em várias etapas: determinação das instruções de máquina da função a ser analisada, do grafo de fluxo de controle [Renaux 93], dos laços e caminhos do grafo, do tempo de execução de cada caminho, e, finalmente, dos limites de tempo de execução da função.

As instruções de máquina da função analisada estão contidas na base de dados gerada a partir dos arquivos-objeto do projeto. O PERF, através da ferramenta e *plugin* Processador, identifica cada instrução de forma que o grafo de fluxo de controle possa ser gerado. Tanto as instruções em *Assembly* quanto o grafo de fluxo de controle podem ser apresentados pelo ambiente (Figura 3).

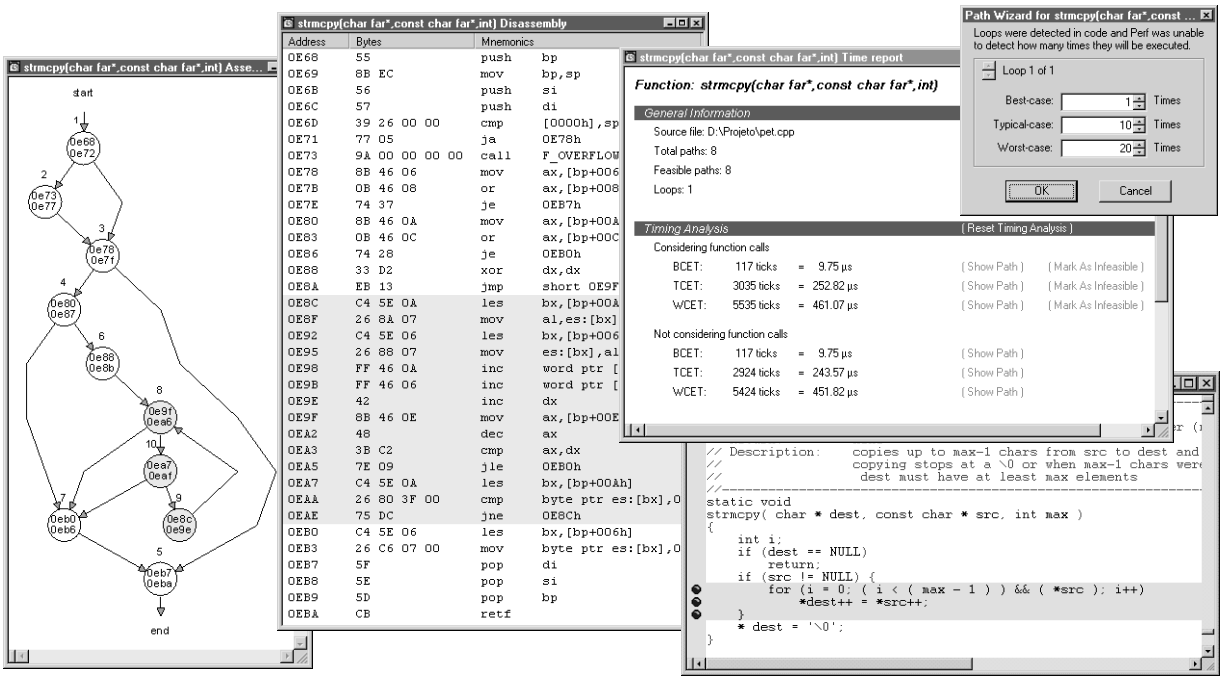

Figura 3 - Função strmcpy representada no PERF. Trechos em cinza evidenciam laços.

Os caminhos do grafo são identificados pelo PERF, que usa uma notação especial para expressar a existência de laços nos caminhos. Por exemplo, considerando a função *strmcpy* da Figura 3, o PERF percorre o grafo de fluxo de controle procurando por todos os caminhos que originam-se no início da função, representado pelo nó *start*, e que terminem em seu nó de saída, representado pelo nó *end*. São encontrados os seguintes caminhos, representados pelo número existente acima de cada nó:

```
caminho:  1  3  5
caminho:  1  3  4  7  5
caminho:  1  3  4  6  8 (10 9 8)  7  5
caminho:  1  3  4  6  8 (10 9 8) 10  7  5
caminho:  1  2  3  5
caminho:  1  2  3  4  7  5
caminho:  1  2  3  4  6  8 (10 9 8)  7  5
caminho:  1  2  3  4  6  8 (10 9 8) 10  7  5
```

O conjunto de nós entre parênteses determina um laço. Caracteriza-se um laço durante a construção de um caminho, quando um nó já inserido no caminho é apontado por outro nó da seqüência.

Este laço pode ser ignorado, ou seja, não ser executado nenhuma vez, ou então pode ser repetido um determinado número de vezes.

Os caminhos necessitam ser resolvidos para que a estimação de tempo possa ocorrer, ou seja, o usuário deve prover informações sobre todos os laços existentes no programa. Estas informações podem ser obtidas por inferência de anotações no código-fonte ou interativamente, e representam o número de iterações possíveis para cada laço. Caso a informação não esteja presente no código-fonte, seja pela inexistência de anotação ou pela inexistência do próprio arquivo-fonte, a informação é pedida para o usuário através de um diálogo interativo (Figura 3). Ao ser mostrado o diálogo, o usuário tem a visualização dos laços no grafo de fluxo de controle, no código *Assembly* e no código-fonte, que fornece subsídio para que o usuário possa determinar o número mínimo, típico e máximo de iterações. Esta abordagem permite que praticamente qualquer tipo de construção, código otimizado e de instruções de salto possam ser utilizados em programas.

Por fim, cada um dos caminhos tem o tempo de execução determinado através do modelo de *hardware*. Com as informações sobre as repetições do caminho e com cada laço existente neste caminho com o número de repetições resolvido, aplica-se o modelo de *hardware* para cada caminho. O modelo de *hardware* analisa cada caminho e faz a soma dos tempos de execução das instruções contidas em cada um de seus nós. O caminho com maior tempo de execução determina o WCET da função, o caminho com menor tempo de execução determina o BCET.

Diversos modelos de *hardware* podem ser utilizados no PERF. Atualmente, dois modelos para o processador 80186 foram desenvolvidos: um modelo simplificado, no qual somente os tempos de execução e os de busca em memória de cada instrução são computados; e o modelo avançado, o qual adiciona o efeito da busca antecipada de instruções e dos conflitos entre busca de instruções e acessos de leitura e escrita de dados no cálculo do tempo.

A ferramenta de estimação agregada ao ambiente PERF apresenta como resultado os menores e maiores tempos possíveis de execução para funções e trechos de funções selecionados pelo usuário. Tempos típicos de execução são calculados a fim de complementar a análise de tempo. Para cada um dos resultados, podem ser consideradas ou não as chamadas de funções. Tempos de execução considerando chamadas determinam a situação real de execução, enquanto tempos de execução desconsiderando-as indicam a influência das instruções da função no tempo total. Além disso, o PERF auxilia o desenvolvedor realçando os caminhos responsáveis pelo maior e menor tempos no grafo de segmentos, no código de máquina e no programa-fonte.

4 IMPLEMENTAÇÃO DO PERF

O projeto PERF foi implementado em C++ utilizando-se o ambiente Microsoft Visual C++ 6.0. Compreende atualmente 75.000 linhas de código distribuídos em 130 classes, aproximadamente.

5 EVOLUÇÃO DO PERF

Pretende-se que o ambiente PERF seja utilizado durante todo o ciclo de desenvolvimento de *software*. Atualmente os ciclos iniciais, compreendendo edição, compilação, ligação e análise estática de tempo de execução, já estão atendidos. Pretende-se agregar ao PERF as seguintes funcionalidades:

- *Ferramenta de medição de tempo*: determinará o tempo de execução de funções através de técnicas de medição.
- *Análise de escalonabilidade*: esta ferramenta fará, a partir dos tempos de execução (medidos e/ou estimados) de cada processo, e a partir do modelo do sistema operacional e do *hardware* do sistema dedicado, a análise do escalonamento dos processos. O sistema poderá ser simulado com diferentes modelos de escalonamento a fim de se procurar a melhor política a ser aplicada. Os resultados que esta ferramenta deverá fornecer são: linhas de tempo, incluindo as mensagens entre processos, tempos de resposta, taxa de utilização do processador pelos processos e verificação formal da escalonabilidade, utilizando para isto a teoria de escalonamento de taxa monotônica (*rate monotonic*).
- *Depuração remota*: permitirá que o programa sendo executado no *target* possa ser depurado remotamente.
- *Gerenciador de projetos*: esta ferramenta gerenciará projetos tendo os objetivos:

- Auxiliar na criação de novos projetos a partir de modelos de projetos pré-existentes.
- Organizar e auxiliar a inclusão de comentários nos cabeçalhos de funções e documentação de módulos de programa.
- Compartilhar os arquivos do projeto entre os programadores de forma que não seja necessário cada programador ter uma cópia em seu computador dos arquivos criados por outros programadores.
- Proteger o código escrito por um programador contra sobrescrita por outro usuário não autorizado.
- Gerenciar as versões e configurações do projeto.

• *Repositório de informações*: permitirá que as informações geradas pelas ferramentas possam ser compartilhadas por vários usuários através de uma rede local. Também armazenará informações sobre todos os projetos efetuados e não somente o que estiver em desenvolvimento. Esta ferramenta substituirá a atual base de dados local.

6 CONCLUSÕES

Um ambiente no qual o processo de teste temporal é integrado ao processo usual de escrita e compilação de programas mostra-se promissor para as atividades de desenvolvimento de *software* para sistemas em tempo real. O ambiente PERF e a ferramenta de estimação de tempo de execução procuram atender esta vertente, fornecendo ao desenvolvedor, de forma interativa, subsídios para que os requisitos temporais sejam testados.

7 REFERÊNCIAS BIBLIOGRÁFICAS

[Cinderella 01] *Página da ferramenta Cinderella*. http://www.ww.princeton.edu/~yauli/cinderella-3.0.

[Góes 01] Góes, J.; Linhares, R.; Renaux, D. *Estimação de Tempo de Execução de Programas no Ambiente PERF*. In Workshop de Tempo Real, do XIX Simpósio Brasileiro de Redes de Computadores. Florianópolis: 2001. http://lit.cpdtt.cefetpr.br.

[Góes 01a] Góes, J. *PERF: Ambiente de Desenvolvimento e Estimação Temporal de Sistemas em Tempo Real*. Dissertação de Mestrado no Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial. CEFET-PR. Curitiba: 2001. http://lit.cpdtt.cefetpr.br.

[Engblom 99] Engblom, J.; Ermedahl, A.; Sjödin, M.; Gustafsson, J; Hansson, H. *Towards Industry-Strength Worst-Case Execution Time Analysis*. ASTEC Technical Report 99/02 and DoCS Technical report 99/109.

[Kawamura 99] Kawamura, A. *Análise Estática de Programas para Medição de Tempos de Execução*. Dissertação de Mestrado no Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial. CEFET-PR. Curitiba-PR.

[Li 95] Li, Y.; Malik, S. *Performance Analysis of Embedded Software Using Implicit Path Numeration.* In Proceedings of the ACM SIGPLAN Workshop on Languages, Compilers and Tools for Real-Time Systems. La Jolla, USA. p. 95-105.

[Lim 94] Lim, S.; Bea, Y.; Jang, G.; Rhee, B.; Min, S.; Park, C.; Shin, H.; Kim, C. *An Accurate Worst Case Timing Analysis for RISC Processors*. In Proceedings of 15th Real-Time Systems Symposium. p. 97-108.

[Park 91] Park, C.; Shaw, A. *Experiments with a Program Timing Tool Based on Source-Level Timing Schema*. IEEE Computer, v. 24, n. 5, p. 48-57. Maio 1991.

[Puschner 00] Puschner, P.; Burns A. *A Review of Worst-case Execution-Time Analysis (Editorial).* Technische Universität Wien. Real-Time Systems, v. 18, n. 2/3, p. 115-128, Maio 2000.

[Renaux 93] Renaux, D. RTX-Parlog: *A Concurrent Logic Programming Language for Real-Time Systems*. Waterloo, Canada, 1993. Ph.D. Thesis. University of Waterloo.

[Renaux 99] RENAUX, Douglas P. B.; BRAGA, André Schinzel; KAWAMURA, Alexandre. *PERF3: Um ambiente para avaliação temporal de Sistemas em Tempo Real* in: II Workshop de Sistemas em Tempo Real. Salvador: 1999, p. 76-87.

[Stoyenko 91] Stoyenko, A.; Hamacher, V.; Holt, R. *Analysing Hard-Real-Time Programs for Guaranteed Schedulability*. IEEE Transactions on Software Engineering v. 17, n. 8, p. 737-750.